Abordagem Semiautomática para Transformação de Documentos Normativos em Requisitos de Software baseada em LLM, RAG e Engenharia de Prompt

Flávio Henrique de Oliveira
Departamento de Computação, Universidade Estadual de Londrina
Londrina, Brasil
flavio.henrique.pos@uel.br

André Luís Andrade Menolli
Departamento de Computação, Universidade Estadual de Londrina
Londrina, Brasil
andremenolli@uel.br

Resumo

A elicitação de requisitos a partir de documentos normativos ainda depende fortemente da interpretação de analistas. Em ambientes regulados, esse processo é crítico, pois normas, regulamentos e políticas internas podem conter incompletudes, inconsistências e contradições que, se não identificadas, podem gerar requisitos incompletos, pouco verificáveis ou desalinhados às regras do sistema. Este artigo apresenta uma abordagem semiautomática para transformar documentos normativos em requisitos de software, combinando Large Language Models, Retrieval-Augmented Generation e Engenharia de Prompt. A principal contribuição está em tratar o documento normativo não apenas como fonte de informação, mas também como objeto de verificação antes da geração dos requisitos. A abordagem recupera trechos relevantes, identifica regras e exceções e gera artefatos de requisitos de software com rastreio explícito das fontes normativas. A pesquisa foi conduzida com base em Design Science Research e avaliada em regulamentos de bibliotecas em língua portuguesa, considerando o caso de uso Realizar Empréstimo de Livro. Os resultados preliminares indicam que a abordagem foi capaz de gerar cenários de caso de uso com referências normativas explícitas, permitindo ao analista verificar a origem das regras representadas e avaliar a aderência dos requisitos ao documento de origem.

Palavras-chave

Engenharia de Requisitos, LLM, RAG, Documentos Normativos, Rastreio

1 Introdução

A transformação de regras institucionais, legais e regulatórias em software é um desafio recorrente em organizações públicas e privadas. Regulamentos, normas, portarias e editais definem obrigações, restrições, exceções, prazos e condições que precisam ser corretamente interpretados para orientar o desenvolvimento de software. Quando essa interpretação ocorre de forma manual, o processo tende a ser demorado, dependente de especialistas e sujeito a falhas, podendo gerar retrabalho, inconsistências, sistemas em não conformidade e prejuízos operacionais.

Nesse contexto, a Engenharia de Requisitos (ER) desempenha papel central na Engenharia de Software, pois decisões tomadas nessa etapa influenciam diretamente o projeto, a implementação e a validação do sistema. Falhas de entendimento, omissões ou especificações inadequadas aumentam riscos de retrabalho e comprometem a qualidade do software [15]. A ER pode ser compreendida como o processo de identificar o propósito do sistema, reconhecer os *stakeholders* envolvidos e documentar suas necessidades para apoiar análise, comunicação e implementação [14].

O problema torna-se mais crítico em domínios regulados, nos quais a conformidade é uma condição necessária para a operação segura e confiável dos sistemas. Nesses ambientes, os requisitos precisam ser derivados de documentos normativos desde as fases iniciais da ER, a fim de reduzir retrabalho, evitar interpretações divergentes e mitigar riscos de não conformidade ao longo do ciclo de vida do software [13]. No entanto, documentos normativos frequentemente apresentam linguagem jurídica densa, referências cruzadas, dependência de contexto e estruturas hierárquicas complexas, o que dificulta a extração consistente e rastreável de regras relevantes para o sistema [23].

As *Large Language Models* (LLMs) vêm sendo investigadas como apoio à leitura, interpretação e transformação de textos complexos em artefatos de Engenharia de Software. Estudos recentes indicam potencial em atividades de elicitação, análise, especificação e validação de requisitos, mas também apontam limitações importantes em contextos críticos, especialmente quanto à confiabilidade, à interpretabilidade, ao rastreio e à geração de conteúdo plausível, porém não factual [4, 6, 12]. Assim, sua adoção em ambientes regulados exige estratégias que obriguem o modelo a usar evidências recuperadas, indicar fontes normativas e evitar inferências não sustentadas pelo documento.

Uma forma de incorporar documentos normativos ao processo de geração é utilizar *Retrieval-Augmented Generation* (RAG), que permite ancorar a geração em evidências recuperadas de uma base documental. Essa estratégia reduz a dependência exclusiva do conhecimento paramétrico do modelo e favorece maior aderência às fontes utilizadas [3, 17]. Em regimes regulatórios, essa característica é relevante, pois permite recuperar trechos normativos específicos e explicitar a base textual que sustenta os artefatos gerados.

Apesar dos avanços no uso de LLMs na ER, inclusive em tarefas de rastreio apoiadas por RAG [7, 11], ainda há lacunas importantes no domínio regulatório. Muitas abordagens dependem de supervisão humana intensa para assegurar aderência estrita ao texto normativo [1], enquanto a associação precisa e auditável entre requisito gerado e base legal permanece como desafio em aberto [5]. Além disso, a carência de *pipelines* integrados e de avaliações replicáveis limita a confiança técnica necessária para aplicações de conformidade regulatória [4, 26].

Este trabalho propõe e avalia uma abordagem semiautomática para transformar documentos normativos em artefatos de requisitos de software. A proposta combina LLMs, RAG e Engenharia de Prompt para recuperar trechos relevantes, estruturar regras regulatórias, gerar requisitos funcionais e explicitar informações ausentes que possam comprometer a implementação correta do sistema. O diferencial da abordagem está em tratar o documento normativo não apenas como fonte de consulta, mas também como objeto de verificação antes da geração dos requisitos.

A avaliação foi conduzida em regulamentos de bibliotecas, considerando o caso de uso *Realizar Empréstimo de Livro*. Esse domínio foi utilizado como cenário inicial de validação por apresentar regras normativas comparáveis e suficientemente estruturadas para analisar a qualidade dos artefatos gerados. Embora o estudo esteja delimitado a regulamentos de bibliotecas, a abordagem pode ser adaptada a outros contextos regulados, como editais, normas institucionais, políticas internas, regulamentos acadêmicos e documentos de conformidade organizacional.

Este trabalho contribui com: (i) uma abordagem semiautomática baseada em RAG e LLMs para derivação de requisitos a partir de regulamentos; (ii) um esquema de geração controlada com rastreio normativo explícito; (iii) uma avaliação empírica em regulamentos de bibliotecas com comparação entre GPT-4o e GPT-5.2. Como contribuição para a trilha de inovação, o estudo apresenta um fluxo semiautomático que combina recuperação de evidências, geração controlada e rastreio normativo para apoiar organizações que precisam derivar requisitos a partir de documentos regulatórios.

2 Trabalhos Relacionados

O uso de LLMs em Engenharia de Requisitos tem sido investigado em tarefas como elicitação, classificação, refinamento, rastreio e geração de artefatos. Esses estudos indicam potencial para reduzir esforço manual, mas também evidenciam a necessidade de mecanismos de controle sobre as saídas geradas. Em documentos normativos, o problema é mais crítico, pois normas e regulamentos podem conter lacunas, ambiguidades, incoerências e contradições que afetam diretamente a qualidade dos requisitos derivados. Esta seção discute trabalhos próximos ao uso de LLMs na ER, à extração de requisitos e à análise de conformidade, destacando a lacuna explorada neste artigo: tratar o documento normativo como objeto de verificação antes da geração dos requisitos.

Ronanki et al. investigam o uso de IA generativa em Engenharia de Requisitos por meio de padrões de prompt, com foco em classificação e rastreamento de requisitos. O estudo mostra que a formulação do prompt influencia a qualidade das respostas, reforçando a Engenharia de Prompt como mecanismo de controle da geração [19]. Rahman e Zhu propõem a GeneUS, ferramenta baseada em GPT-4 para gerar histórias de usuário e casos de teste a partir de documentos de requisitos [16]. Embora ambos contribuam para automatizar atividades da ER, partem de requisitos ou documentos já estruturados, sem aprofundar a análise prévia de documentos normativos como fonte primária de regras, exceções, incompletudes e conflitos.

Em contextos regulados, Hassani explora o uso de LLMs para extração de conteúdo legal relacionado a requisitos e verificação de conformidade, especialmente em segurança alimentar e artefatos associados à GDPR [9]. O trabalho se aproxima desta pesquisa por lidar com textos legais e regulatórios, mas concentra-se na identificação de conteúdo jurídico e na checagem de conformidade. De forma complementar, o ReqNet combina LLMs e arquiteturas de aprendizado profundo para extrair requisitos de documentos não estruturados [20]. Apesar da relevância, essa abordagem está mais associada à identificação de sentenças candidatas a requisitos do que à interpretação normativa de regras, exceções, prazos, sanções e lacunas que podem comprometer a geração posterior dos requisitos.

Ferrari e Spoletini discutem a relação entre Engenharia de Requisitos formal e LLMs, destacando preocupações com corretude, justiça e confiabilidade dos artefatos gerados [6]. Essa discussão é relevante para documentos normativos, pois uma interpretação incorreta pode gerar impactos técnicos, jurídicos e operacionais. A literatura recente também tem investigado a detecção de contradições e inconsistências em textos regulatórios. Schumann e Gómez analisam documentos regulatórios alemães com modelos supervisionados aplicados a pares de sentenças [21], enquanto Schumann, Gómez e Pargmann investigam uma abordagem baseada em Engenharia de Prompt para a mesma tarefa [22]. Gärtner e Göhlich, por sua vez, propõem o ALICE, que combina LLMs e lógica formal para detectar contradições em requisitos escritos em linguagem natural controlada [8]. Esses estudos mostram que inconsistências podem ser tratadas computacionalmente, mas ainda se concentram na detecção do problema, não em sua integração com a geração de requisitos.

3 Metodologia

Esta pesquisa foi conduzida com base na abordagem de *Design Science Research* (DSR), adequada para estudos que envolvem a construção, aplicação e avaliação de artefatos destinados à solução de problemas práticos em contextos organizacionais e tecnológicos [10]. A pesquisa teve como objetivo projetar e avaliar uma abordagem semiautomática capaz de apoiar a transformação de documentos normativos em requisitos escritos em cenários de caso de uso. Para orientar a condução do estudo, foram definidas as seguintes questões de pesquisa:

- (RQ1) Como uma abordagem semiautomática baseada em RAG, LLMs e Engenharia de Prompt pode apoiar a transformação de documentos normativos em requisitos funcionais de software?
- (RQ2) Em que medida os requisitos gerados preservam o significado das regras normativas e permanecem alinhados às obrigações, restrições e exceções presentes no documento de origem?
- (RQ3) Como os artefatos gerados se comportam em relação aos critérios de qualidade de requisitos?

O estudo foi realizado com seis regulamentos de bibliotecas em língua portuguesa. Esses documentos foram escolhidos por apresentarem regras comparáveis sobre empréstimo de materiais bibliográficos e por conterem condições, exceções e restrições típicas de um domínio regulado. Em todos os documentos, foi analisado o mesmo caso de uso: Realizar Empréstimo de Livro.

O protocolo foi aplicado da mesma forma em todos os documentos. Primeiro, cada regulamento foi processado e indexado na base vetorial. Depois, foram executadas consultas semânticas para

recuperar os trechos relacionados ao empréstimo. Em seguida, o contexto recuperado foi fornecido à LLM para geração do caso de uso, contendo fluxo principal, fluxos alternativos, exceções e referências normativas. Por fim, os artefatos gerados foram avaliados manualmente segundo critérios de qualidade de requisitos.

Foram comparados os modelos GPT-4o, identificado na API como `gpt-4o`, e GPT-5.2, identificado como `gpt-5.2`. Para a representação vetorial dos fragmentos, foi utilizado o modelo de embeddings `text-embedding-3-small`. Nas execuções com o modelo `gpt-5.2`, o esforço de raciocínio foi configurado como `high`, conforme os parâmetros suportados pelo modelo. Os documentos, as consultas, o processo de recuperação, o prompt e o formato de saída foram mantidos constantes entre as execuções.

Todos os regulamentos foram processados com a mesma configuração de RAG. O texto foi segmentado preservando marcadores estruturais, como artigos, seções, anexos e itens, em fragmentos de até 4.500 caracteres, com sobreposição de 700 caracteres. A recuperação utilizou 27 consultas semânticas e similaridade de cosseno, selecionando os 20 fragmentos mais relevantes por consulta. Após a deduplicação, foram adicionados fragmentos das extremidades e distribuídos ao longo do documento, com cobertura aproximada de 70 fragmentos. O contexto final foi limitado a 160.000 caracteres, mantendo-se essa configuração nos seis regulamentos.

A unidade de análise adotada foi cada passo produzido no fluxo do caso de uso. Os artefatos gerados foram avaliados segundo os critérios de ambiguidade, corretude, completude e rastreio, com base em métricas de qualidade para requisitos discutidas na literatura [2]. A ambiguidade foi observada quando um passo permitia mais de uma interpretação possível. A corretude foi analisada pela aderência semântica entre o requisito gerado e o conteúdo normativo correspondente. A completude foi avaliada pela cobertura das ações, condições, exceções e restrições relevantes ao processo analisado. O rastreio foi verificado pela presença de referência normativa explícita, válida e compatível com o trecho recuperado.

Em que P_{sem_amb} representa a quantidade de passos não ambíguos, P_{aval} os passos avaliados, P_{corr} os passos corretos, P_{ger} os passos gerados, E_{cont} os elementos normativos contemplados, E_{esp} os elementos esperados e P_{fonte} os passos com fonte normativa válida.

$$NA = \frac{P_{sem\_amb}}{P_{aval}}, \qquad COR = \frac{P_{corr}}{P_{ger}}, \qquad COMP = \frac{E_{cont}}{E_{esp}}, \qquad RAST = \frac{P_{fonte}}{P_{ger}}. \tag{1}$$

A etapa de refinamento humano, prevista na abordagem, não foi avaliada nesta fase do estudo. A avaliação concentrou-se nos artefatos gerados a partir do documento normativo e do contexto recuperado pelo RAG. Essa delimitação permite separar os resultados já obtidos dos componentes ainda em evolução. Em trabalhos futuros, pretende-se avaliar se as perguntas geradas pela LLM para o analista, e suas respectivas respostas, melhoram a corretude, a completude, o rastreio e a redução de ambiguidades dos requisitos.

4 Abordagem

Conforme sintetizado na Figura 1, o processo é composto por quatro etapas principais. Na primeira, o documento normativo em PDF é submetido à extração e limpeza textual, com remoção de ruídos, cabeçalhos e quebras artificiais. Em seguida, o texto é fragmentado por uma estratégia híbrida, que preserva a estrutura normativa do documento, como artigos e seções, e mantém sobreposição entre trechos para reduzir perda de contexto. Os fragmentos são então convertidos em *embeddings* e armazenados em uma base vetorial.

Na segunda etapa, ocorre a recuperação semântica dos trechos normativos. O analista define o escopo funcional a ser analisado e os critérios de busca. Neste estudo, o escopo corresponde ao caso de uso *Realizar Empréstimo de Livro*. A partir desse escopo, o sistema recupera trechos relevantes por similaridade de cosseno, remove duplicações e monta o contexto RAG, composto por evidências textuais, referências de origem e pontuações de recuperação.

A terceira etapa corresponde à análise normativa e ao refinamento dos requisitos. Com base no contexto recuperado, a LLM identifica inconsistências, contradições e incompletudes do documento normativo. Quando uma informação necessária não está explícita ou apresenta conflito, o modelo gera perguntas objetivas para o analista. As respostas fornecidas são incorporadas ao contexto, formando uma base refinada para a geração dos artefatos.

Na quarta etapa, a LLM gera os cenários de caso de uso a partir do contexto refinado. A saída inclui fluxo principal, fluxos alternativos, exceções, referências normativas associadas aos passos e um relatório de auditoria com dúvidas, decisões e evidências utilizadas. Esses artefatos passam por revisão técnica do analista de requisitos. Caso sejam identificados problemas, o feedback retorna ao processo para novo refinamento. Quando aprovados, os artefatos compõem o pacote de requisitos homologado.

Os *prompts* foram definidos com instruções explícitas sobre o papel do modelo, o contexto da tarefa, o formato esperado da saída e a obrigatoriedade de vincular cada elemento gerado à respectiva fonte normativa. Essa estrutura segue diretrizes de Engenharia de Prompt relacionadas a *Context*, *Persona*, *Templates*, *Disambiguation* e *Wording* [18], além de recomendações para delimitar a tarefa, impor restrições e controlar o formato da resposta [24, 25].

5 Resultados Preliminares

A avaliação inicial concentrou-se nas seguintes etapas: construção da base de conhecimento, recuperação semântica dos trechos normativos e geração dos artefatos de requisitos. Portanto, nesta fase, a etapa de refinamento humano ainda não foi avaliada. Essa etapa foi incluída após a análise dos resultados. Observou-se que problemas presentes nos documentos normativos, como incompletudes, incoerências e contradições, também apareciam nos casos de uso gerados pela LLM. Quando a norma não apresentava uma regra de forma clara ou completa, o requisito produzido podia ficar incompleto, ambíguo ou dependente de interpretação. Deste modo, a geração automática mostrou-se útil para gerar os requisitos e indicar suas fontes normativas, mas não substitui a revisão do analista. O refinamento humano é necessário para interpretar lacunas, avaliar conflitos e validar requisitos quando o documento de origem não fornece informação suficiente.

Com base nos resultados iniciais, a abordagem passou a considerar uma etapa em evolução, voltada à identificação de problemas no próprio documento normativo. Nessa etapa, a LLM, orientada por Engenharia de Prompt, deve detectar possíveis incompletudes,

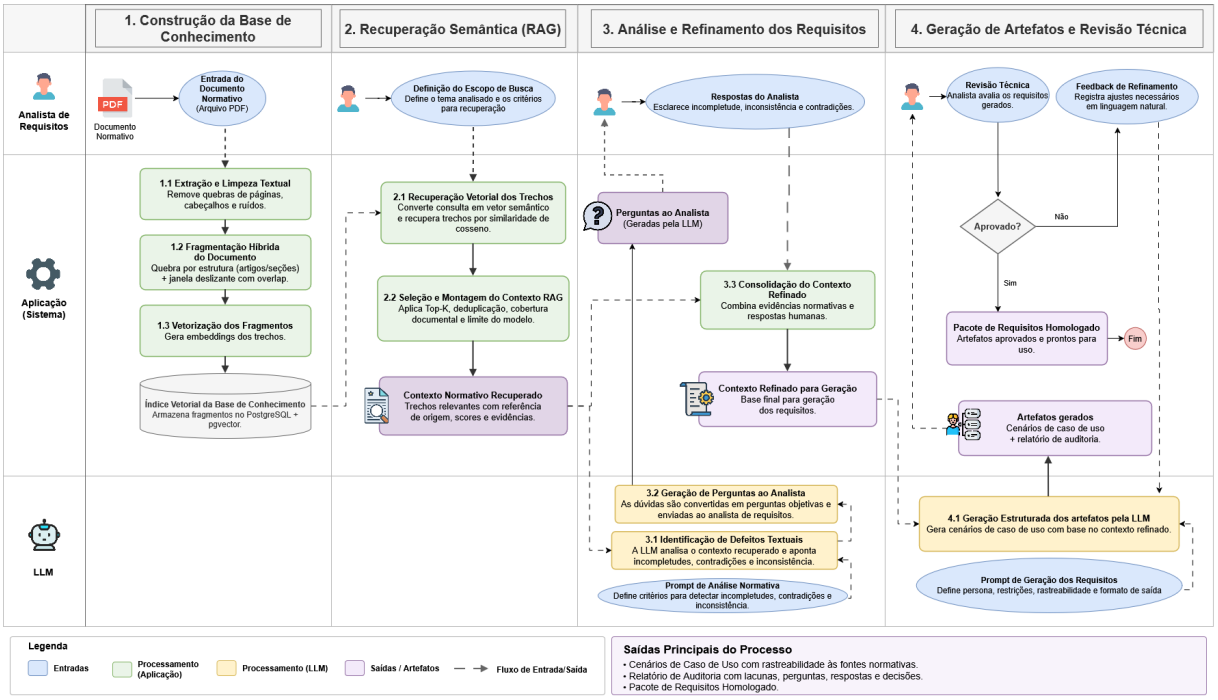

Figura 1: Arquitetura da abordagem proposta.

inconsistências e contradições que possam comprometer a geração dos requisitos, em alinhamento com estudos recentes sobre detecção automatizada de inconsistências e contradições em documentos regulatórios e requisitos em linguagem natural [8, 21, 22]. O modelo deve transformá-los em perguntas objetivas para o analista de requisitos. O papel do analista é responder a essas perguntas ou buscar os esclarecimentos necessários junto aos responsáveis pela norma. Com essas respostas, uma nova versão dos requisitos pode ser produzida. Em etapa posterior da pesquisa, pretende-se comparar os artefatos gerados antes e depois desse processo, a fim de verificar se as respostas do analista contribuem para melhorar a corretude, a completude, o rastreio e a redução de ambiguidades dos requisitos.

A Tabela 1 apresenta um trecho do caso de uso *Realizar Empréstimo de Livros*, gerado automaticamente pelo GPT-5.2 a partir de um dos regulamentos analisados. O exemplo mostra que a LLM conseguiu transformar regras normativas em um artefato estruturado de Engenharia de Requisitos, organizando o comportamento esperado do sistema em fluxo principal e fluxos alternativos. Também se observa a indicação explícita das fontes normativas utilizadas, o que permite ao analista verificar a origem das decisões representadas no artefato.

No fluxo principal, o modelo organizou as condições centrais do processo de empréstimo, incluindo a solicitação dos livros, a validação do cadastro ativo, a verificação de restrições impeditivas, o controle do limite de empréstimos simultâneos, o registro da operação e a definição do prazo de devolução. Nos fluxos alternativos, foram representadas situações de exceção, como cadastro inativo e ultrapassagem do limite permitido de livros. Esse resultado sugere

Tabela 1: Trecho do caso de uso gerado por GPT-5.2

Caso de Uso: Realizar Empréstimo de Livros

Fluxo Principal

1. Usuário solicita o empréstimo de 1 a 3 livros, identificando os exemplares desejados, estando com cadastro ativo na Biblioteca [Fonte: Art. 15; Art. 16].
2. Sistema valida que o cadastro do usuário está ativo para habilitar o empréstimo [Fonte: Art. 15].
3. Sistema valida que o usuário não possui restrições decorrentes de violações de quaisquer disposições do regimento interno, condição impeditiva para empréstimo [Fonte: Art. 15].
4. Sistema valida que a quantidade total de livros em empréstimo simultâneo para o usuário, incluindo o novo pedido, não excede 3 livros [Fonte: Art. 16].
5. Sistema registra o empréstimo dos livros e define o prazo de devolução de cada exemplar para o período máximo de 8 dias corridos [Fonte: Art. 16].
6. Sistema efetiva o empréstimo sem cobrança de taxa pelo uso do serviço oferecido pela Biblioteca [Fonte: Art. 7].

Fluxos Alternativos

Exceção: Cadastro não está ativo

1. Sistema identifica que o usuário não possui cadastro ativo na Biblioteca, condição necessária para empréstimo [Fonte: Art. 15].
2. Sistema nega o empréstimo por ausência de cadastro ativo [Fonte: Art. 15].

Exceção: Limite de 3 livros simultâneos excedido

1. Sistema identifica que a operação resultaria em mais de 3 livros emprestados simultaneamente pelo usuário [Fonte: Art. 16].
2. Sistema nega o empréstimo de forma a manter o limite de até 3 livros por vez [Fonte: Art. 16].

que a combinação entre recuperação semântica e geração controlada por *prompt* pode apoiar a organização inicial de requisitos funcionais derivados de documentos normativos, desde que acompanhada por revisão humana nas situações em que a própria norma apresenta lacunas ou conflitos.

A avaliação foi conduzida com base nos indicadores de qualidade propostos por Bokhari e Siddiqui [2]. A ambiguidade foi avaliada

pela contagem dos passos gerados que admitiam mais de uma interpretação possível em relação ao total de passos analisados. A corretude foi medida pela proporção de elementos gerados em conformidade com as regras efetivamente previstas no regulamento, considerando-se corretos aqueles semanticamente aderentes ao texto normativo. A completude foi verificada pela razão entre a quantidade de ações, condições e exceções relevantes do processo normativo contempladas na saída e o total de elementos esperados a partir do regulamento. O rastreio, por sua vez, foi medido pela proporção de unidades geradas que apresentavam referência normativa explícita, válida e compatível com o trecho efetivamente recuperado pelo RAG. Dessa forma, a metodologia articula a proposição do artefato, sua demonstração em documentos reais e uma avaliação empírica orientada por indicadores de qualidade em Engenharia de Requisitos.

Tabela 2: Comparação entre os modelos GPT-4o e GPT-5.2

Documento	Não Amb.		Corretude		Completude		Rastreio	
	4o	5.2	4o	5.2	4o	5.2	4o	5.2
1-BPBL-MA	71,87	85,71	78,12	85,71	88,89	100	87,5	100
2-SESI/SENAI	91	100	70	100	60	88,86	60	92
3-SIBFESFX	83,33	94,29	60	85	50	91	80	100
4-BPMST-SC	96	100	92,31	100	85	98	100	100
5-TUIUTI	87,50	95,65	81,25	90	56,67	95,86	81,25	100
6-UTFPR	87,50	100	71,88	100	58,33	100	75	100

Os resultados da Tabela 2 mostram que o GPT-5.2 apresentou desempenho superior ao GPT-4o em todos os documentos e indicadores avaliados. De modo geral, o GPT-5.2 alcançou valores mais altos de não ambiguidade, corretude, completude e rastreio, com destaque para os ganhos mais expressivos em completude e rastreio. Esse comportamento sugere maior capacidade do modelo em recuperar, integrar e explicitar condições, exceções e referências normativas de forma consistente. As diferenças foram mais acentuadas em documentos como UTFPR, SIBFESFX e SESI/SENAI, indicando que o GPT-4o tende a ser mais sensível a regulamentos mais extensos, densos ou menos uniformes. Em contraste, no documento BPMST-SC, ambos os modelos apresentaram desempenho elevado, o que indica que textos normativos mais objetivos e estruturados tendem a reduzir a diferença entre os modelos.

Sob a perspectiva econômica, conforme a Tabela 3, os resultados evidenciam um *trade-off* entre qualidade e custo operacional. Embora o GPT-5.2 tenha produzido artefatos mais corretos, completos e rastreáveis, seu tempo de execução e custo por documento foram substancialmente superiores aos do GPT-4o. Isso indica que o GPT-4o pode ser mais adequado em cenários com restrição orçamentária ou necessidade de respostas rápidas, enquanto o GPT-5.2 se mostra mais apropriado para contextos em que a qualidade e a auditabilidade são prioritárias. Do ponto de vista social, a melhoria em rastreio e redução de ambiguidades é relevante porque aumenta a transparência da interpretação normativa, reduz riscos de implementação incorreta e fortalece a confiabilidade de sistemas que impactam direitos, deveres e acesso a serviços em domínios regulados.

Tabela 3: Comparação do tempo de execução e do custo por documento entre os modelos GPT-4o e GPT-5.2

Documento	Tempo 4o (s)	Tempo 5.2 (s)	Custo 4o (USD)	Custo 5.2 (USD)
1-BPBL-MA	30	520	0.35	1.80
2-SESI/SENAI	60	640	0.42	2.00
3-SIBFESFX	90	760	0.50	2.20
4-BPMST-SC	35	440	0.30	2.00
5-TUIUTI	40	560	0.38	1.50
6-UTFPR	70	680	0.45	1.90

5.1 Análise dos Artefatos Gerados

A avaliação concentrou-se nas etapas já implementadas do pipeline: construção da base de conhecimento, recuperação semântica e geração dos artefatos de requisitos. A etapa de refinamento humano, embora prevista na abordagem, não foi avaliada nesta fase. Assim, os resultados apresentados dizem respeito aos artefatos gerados a partir do contexto recuperado pelo RAG e do prompt estruturado.

Os resultados indicam desempenho superior do GPT-5.2 em todos os indicadores avaliados. Considerando a média dos seis regulamentos, o modelo aumentou a taxa de não ambiguidade de 86,20% para 95,94%, a corretude de 75,59% para 93,45%, a completude de 66,48% para 95,62% e o rastreio de 80,62% para 98,67%. Os maiores ganhos ocorreram em completude e rastreio, sugerindo maior capacidade do modelo em integrar regras distribuídas no documento e associar os passos gerados às respectivas fontes normativas.

A diferença entre os modelos foi mais evidente em regulamentos com maior densidade normativa ou com regras distribuídas em diferentes seções, como UTFPR, SIBFESFX e SESI/SENAI. Nesses casos, o GPT-4o apresentou maior dificuldade em cobrir exceções, restrições e consequências associadas ao empréstimo, enquanto o GPT-5.2 produziu artefatos mais completos e com referências normativas mais consistentes. Em documentos mais objetivos e estruturados, como BPMST-SC, a diferença entre os modelos foi menor, indicando que a clareza da fonte normativa influencia diretamente a qualidade dos requisitos gerados.

A análise qualitativa mostrou que a ambiguidade tende a diminuir quando o regulamento apresenta regras operacionais fechadas, com quantidade, prazo, condição de elegibilidade e exceção explicitamente definidos. Em contrapartida, expressões genéricas, regras incompletas ou decisões deixadas "a critério da biblioteca" aumentaram a necessidade de interpretação e resultaram em passos mais sujeitos a ambiguidade. Esse resultado reforça que parte dos problemas encontrados nos requisitos não decorre apenas do modelo, mas da própria qualidade da fonte normativa.

Em relação à corretude, os melhores resultados ocorreram quando os passos gerados permaneceram semanticamente próximos dos dispositivos normativos recuperados. A completude foi maior quando o artefato contemplou não apenas o fluxo principal de empréstimo, mas também restrições, renovações, reservas, atrasos, bloqueios e penalidades. Já o rastreio dependeu da precisão da referência utilizada: citações genéricas ou associadas a dispositivos normativos amplos reduziram a qualidade do rastreio, mesmo quando o conteúdo do passo estava parcialmente correto.

Os resultados também evidenciam um trade-off operacional. O GPT-5.2 apresentou melhor qualidade, mas exigiu maior tempo e custo médio por documento. Enquanto o GPT-4o registrou tempo médio de 54 segundos e custo médio de US$0,40 por documento, o GPT-5.2 registrou tempo médio de 600 segundos e custo médio de US$1,90. Assim, o GPT-4o pode ser mais adequado para análises exploratórias ou cenários com restrição de custo, enquanto o GPT-5.2 se mostra mais apropriado quando completude, corretude e auditabilidade são prioritárias.

6 Limitações de Pesquisa

Este estudo apresenta resultados preliminares e possui limitações que devem ser consideradas. A avaliação foi realizada em um único domínio, com regulamentos de bibliotecas em língua portuguesa, e considerou apenas um caso de uso específico, o que limita a generalização dos resultados para outros documentos normativos. Além disso, a avaliação dos artefatos foi manual e não contou, nesta fase, com múltiplos avaliadores, medidas de concordância ou testes estatísticos, podendo haver subjetividade na análise de ambiguidade, corretude, completude e rastreio. Também foram comparados apenas dois modelos de linguagem, GPT-4o e GPT-5.2, sob uma configuração específica de RAG, prompt e geração, sem variações sistemáticas de parâmetros, estratégias de recuperação ou modelos abertos. Por fim, a etapa de refinamento humano prevista na abordagem ainda não foi avaliada empiricamente, o que limita a análise de seu impacto na qualidade final dos requisitos gerados. Essa etapa será avaliada em trabalhos futuros.

7 Trabalhos Futuros

Como trabalhos futuros, pretende-se avaliar a abordagem com outros modelos de linguagem, diferentes configurações de RAG e variações de *prompt*. Também será analisada empiricamente a etapa de refinamento humano, comparando os requisitos gerados antes e depois das respostas do analista. Além disso, a abordagem será aplicada a outros documentos normativos, como leis, editais, políticas internas e normas de *compliance*, permitindo investigar sua utilização na construção de bases de conhecimento organizacionais e na verificação de conformidade entre requisitos, obrigações legais e procedimentos internos.

8 Conclusão

Em relação à RQ1, verificou-se que a combinação entre construção da base de conhecimento, recuperação semântica e geração controlada por *prompt* pode apoiar a transformação de documentos normativos em requisitos, desde que acompanhada por validação humana. A abordagem mostrou potencial para estruturar regras complexas em cenário de caso de uso, associando cada passo às fontes normativas utilizadas. Contudo, os resultados também mostram que a qualidade final dos requisitos depende tanto do modelo utilizado quanto da coerência, completude e consistência do regulamento analisado.

Quanto à RQ2, observou-se que os requisitos gerados preservam melhor o significado das regras normativas quando o documento apresenta regras claras, objetivas e completas. Nesses casos, os artefatos mantiveram maior aderência às obrigações, restrições e exceções previstas no regulamento. Por outro lado, quando a norma continha informações incompletas, incoerentes ou contraditórias, essas limitações também se refletiam nos requisitos gerados, reforçando a necessidade de validação pelo analista.

Em resposta à RQ3, os artefatos foram avaliados segundo os critérios de ambiguidade, corretude, completude e rastreio. O GPT-5.2 apresentou desempenho superior ao GPT-4o em todos os documentos analisados, com ganhos mais evidentes em completude e rastreio. Esse resultado sugere maior capacidade de integrar regras distribuídas no texto normativo e justificar os requisitos por meio de referências explícitas. Entretanto, o ganho de qualidade veio acompanhado de maior custo e maior tempo de execução, indicando um *trade-off* relevante para adoção prática da abordagem.

Os resultados deste estudo permitem delimitar melhor a contribuição em relação aos trabalhos correlatos. Ronanki et al. mostram que a construção do *prompt* influencia a qualidade das respostas em tarefas de Engenharia de Requisitos [19]. Nesta pesquisa, esse ponto foi confirmado, mas com uma diferença importante: o *prompt* não atua sozinho. Ele é usado para orientar a LLM a operar sobre evidências recuperadas pelo RAG, exigir saída estruturada e vincular cada requisito à fonte normativa correspondente. Assim, enquanto o RAG fornece o contexto documental que fundamenta a geração, a Engenharia de Prompt define como esse contexto deve ser interpretado e transformado em artefatos de requisitos. Rahman e Zhu demonstram que LLMs podem gerar artefatos a partir de documentos de requisitos [16]; porém, os resultados deste estudo indicam que documentos normativos exigem uma etapa anterior de análise da própria fonte, pois contradições, inconsistências e incompletudes presentes na norma tendem a aparecer também nos requisitos gerados. Hassani aproxima-se desse problema ao tratar textos legais e conformidade regulatória [9], enquanto o ReqNet contribui para a extração de requisitos em documentos não estruturados [20]. No entanto, esses trabalhos não integram, em um mesmo fluxo, a recuperação de evidências normativas, a verificação da qualidade da fonte, a geração de requisitos e o rastreio passo a passo. Os achados também reforçam a preocupação de Ferrari e Spoletini sobre confiabilidade, justificativa e controle no uso de LLMs em ER [6], pois requisitos sem fonte precisa ou baseados em inferências não sustentadas reduzem a qualidade do artefato produzido.

Os resultados iniciais indicam que a abordagem é promissora para organizar regras normativas em artefatos de requisitos com maior rastreio, desde que acompanhada por revisão humana quando o documento de origem apresentar incompletudes, inconsistências ou contradições.

Como principal contribuição, o estudo evidencia que a geração de requisitos a partir de documentos normativos não deve ser tratada apenas como extração textual. Ao combinar recuperação de evidências, geração estruturada e rastreio normativo, a abordagem permite verificar se os requisitos estão fundamentados em regras explícitas, preservando a necessidade de revisão humana quando a fonte apresentar lacunas ou conflitos.

Disponibilidade de Artefatos

Os artefatos utilizados neste estudo, incluindo os documentos analisados, os prompts e os artefatos gerados, estão disponíveis em: https://github.com/flaviohenrique096/DocumentosNormativos.

Referências

[1] Sallam Abualhaija, Marcello Ceci, Nicolas Sannier, Domenico Bianculli, Salomé Lannier, Martina Siclari, Olivier Voordeckers, and Stanisław Tosza. 2025. LLM-assisted Extraction of Regulatory Requirements: A Case Study on the GDPR. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*. IEEE, Valencia, Spain, 142–154. doi:10.1109/RE63999.2025.00023 Proceedings. ISBN (print): 979-8-3315-2414-2.

[2] Mohammad Ubaidullah Bokhari and Shams Tabrez Siddiqui. 2011. Metrics for Requirements Engineering and Automated Requirements Tools. In *Proceedings of the 5th National Conference; INDIACom-2011: Computing For Nation Development*. Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi, India. March 10–11, 2011. ISBN 978-93-80544-00-7. ISSN 0973-7529.

[3] Andrew Brown, Muhammad Roman, and Barry Devereux. 2025. A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges. *Big Data and Cognitive Computing* 9, 12 (2025). doi:10.3390/bdcc9120320

[4] Haowei Cheng, Jati H. Husen, Yijun Lu, Teeradaj Racharak, Nobukazu Yoshioka, Naoyasu Ubayashi, and Hironori Washizaki. 2026. Generative AI for Requirements Engineering: A Systematic Literature Review. *Software: Practice and Experience* 56, 2 (2026), 141–170. doi:10.1002/spe.70029

[5] Romina Etezadi, Sallam Abualhaija, Chetan Arora, and Lionel Briand. 2026. Classifier or prompt: A case study on legal requirements traceability. *Empirical Software Engineering* 31, 4 (March 2026). doi:10.1007/s10664-026-10827-1

[6] Alessio Ferrari and Paola Spoletini. 2025. Formal requirements engineering and large language models: A two-way roadmap. *Information and Software Technology* 181 (2025), 107697. doi:10.1016/j.infsof.2025.107697

[7] Dominik Fuchß, Tobias Hey, Jan Keim, Haoyu Liu, Niklas Ewald, Tobias Thirolf, and Anne Koziolek. 2025. LiSSA: Toward Generic Traceability Link Recovery through Retrieval-Augmented Generation. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering* (Ottawa, Ontario, Canada) *(ICSE '25)*. IEEE Press, 1396–1408. doi:10.1109/ICSE55347.2025.00186

[8] Alexander Elenga Gärtner and Dietmar Göhlich. 2024. Automated requirement contradiction detection through formal logic and LLMs. *Automated Software Engineering* 31, 49 (2024). doi:10.1007/s10515-024-00452-x

[9] Shabnam Hassani. 2024. Enhancing Legal Compliance and Regulation Analysis with Large Language Models. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, Reykjavik, Iceland, 507–511. doi:10.1109/RE59067.2024.00065

[10] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28, 1 (2004), 75–105. doi:10.2307/25148625

[11] Tobias Hey, Dominik Fuchß, Jan Keim, and Anne Koziolek. 2025. Requirements Traceability Link Recovery via Retrieval-Augmented Generation. In *Requirements Engineering: Foundation for Software Quality: 31st International Working Conference, REFSQ 2025, Barcelona, Spain, April 7–10, 2025, Proceedings* (Barcelona, Spain). Springer-Verlag, Berlin, Heidelberg, 381–397. doi:10.1007/978-3-031-88531-0_27

[12] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems* 43, 2 (Jan. 2025), 1—-55. doi:10.1145/3703155

[13] Oleksandr Kosenkov, Parisa Elahidoost, Tony Gorschek, Jannik Fischbach, Daniel Mendez, Michael Unterkalmsteiner, Davide Fucci, and Rahul Mohanani. 2025. Systematic mapping study on requirements engineering for regulatory compliance of software systems. *Information and Software Technology* 178 (Feb. 2025), 107622. doi:10.1016/j.infsof.2024.107622

[14] Bashar Nuseibeh and Steve M. Easterbrook. 2000. Requirements Engineering: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*. ACM, Limerick, Ireland, 35–46. doi:10.1145/336512.336523

[15] Roger S. Pressman. 2005. *Software Engineering: A Practitioner's Approach* (6 ed.). McGraw-Hill.

[16] Tajmilur Rahman and Yuecai Zhu. 2024. Automated User Story Generation with Test Case Specification Using Large Language Model. arXiv:2404.01558 [cs.SE] doi:10.48550/arXiv.2404.01558

[17] Markus Reuter, Tobias Lingenberg, Ruta Liepina, Francesca Lagioia, Marco Lippi, Giovanni Sartor, Andrea Passerini, and Burcu Sayin. 2025. Towards Reliable Retrieval in RAG Systems for Large Legal Datasets. In *Proceedings of the Natural Legal Language Processing Workshop 2025*. 17–30. doi:10.18653/v1/2025.nllp-1.3

[18] Krishna Ronanki, Simon Arvidsson, and Johan Axell. 2025. Prompt Engineering Guidelines for Using Large Language Models in Requirements Engineering. arXiv:2507.03405 [cs.SE] doi:10.48550/arXiv.2507.03405

[19] Krishna Ronanki, Beatriz Cabrero-Daniel, Jennifer Horkoff, and Christian Berger. 2023. Requirements Engineering using Generative AI: Prompts and Prompting Patterns. *CoRR* abs/2311.03832 (2023). arXiv:2311.03832 doi:10.48550/ARXIV.2311.03832

[20] Summra Saleem, Muhammad Nabeel Asim, and Andreas Dengel. 2026. ReqNet: an LLM-driven computational framework for automated requirements extraction from unstructured documents. *Complex & Intelligent Systems* 12, 1 (2026), 38. doi:10.1007/s40747-025-02143-w

[21] Gerrit Schumann and Jorge Marx Gómez. 2024. Detection of Contradictions and Inconsistencies in German Regulatory Documents. In *2024 6th International Conference on Natural Language Processing (ICNLP)*. 74–84. doi:10.1109/ICNLP60986.2024.10692679

[22] Gerrit Schumann, Jorge Carlos Marx Gómez, and Hergen Pargmann. 2026. Detection of Contradictions and Inconsistencies in Regulatory Documents Using Prompt-Engineering. In *Proceedings of the 59th Hawaii International Conference on System Sciences*. 1766–1775. doi:10.24251/HICSS.2026.211

[23] Mika Sie, Ruby Beek, Michiel Bots, Sjaak Brinkkemper, and Albert Gatt. 2024. Summarizing Long Regulatory Documents with a Multi-Step Pipeline. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, Nikolaos Aletras, Ilias Chalkidis, Leslie Barrett, Cătălina Goanță, Daniel Preoțiuc-Pietro, and Gerasimos Spanakis (Eds.). Association for Computational Linguistics, Miami, FL, USA, 18–32. doi:10.18653/v1/2024.nllp-1.2

[24] Andreas Vogelsang. 2024. From Specifications to Prompts: On the Future of Generative Large Language Models in Requirements Engineering. *IEEE Software* 41, 5 (Sept. 2024), 9–13. doi:10.1109/ms.2024.3410712

[25] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs.SE] https://arxiv.org/abs/2302.11382

[26] Mohammad Amin Zadenoorila, Jacek Dąbrowski, Waad Alhoshan, Liping Zhao, and Alessio Ferrari. 2025. Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review. *arXiv preprint arXiv:2509.11446* (2025). https://arxiv.org/abs/2509.11446